Final @2011-01-17

1 (True or False) and Justify [20 bodov]

Uveďte, či je tvrdenie pravdivé a nanajvýš tromi vetami svoj názor zdôvodnite.

(Úplne správna odpoveď je za 2.5 boda. Správna odpoveď úplne bez zdôvodnenia je za 0.5 boda. Nesprávna odpoveď, ako aj správna odpoveď s úplne nesprávnym zdôvodnením, sú za 0 bodov.)

Q1 Ak je f primitívne rekurzívna, tak g definovaná predpisom $g(n) = 2^{2^{f(n)}}$ je tiež primitívne rekurzívna.

Q2 Ak je rekurzívna funkcia f surjektívna (teda jej oborom hodnôt je celé $\mathbb{N}$), tak existuje rekurzívna funkcia g taká, že $\forall n : f(g(n)) = n$.

Q3 Ak je čiastočne rekurzívna funkcia f surjektívna (teda jej oborom hodnôt je celé $\mathbb{N}$, ale definičným oborom môže byť len podmnožina $\mathbb{N}$), tak existuje (totálna!) rekurzívna funkcia g taká, že $\forall n : f(g(n)) = n$.

Q4 Definičný obor čiastočne rekurzívnej funkcie je čiastočne rekurzívna množina.

Q5 Obor hodnôt čiastočne rekurzívnej funkcie je čiastočne rekurzívna množina.

Q6 Existuje zhora ohraničená rekurzívna funkcia, ktorá nie je primitívne rekurzívna.

Q7 Pod slovom *programy* teraz uvažujme programy v našom Pascale, ktoré počítajú unárne totálne funkcie.

Pekná postupnosť je každá postupnosť programov $P_1, P_2, \ldots$ taká, že P_1 je identita a pre každé $i > 1$ platí: program P_{i+1} má časovú zložitosť $O(P_i(n))$. (Slovne: pre skoro každý vstup n nám program P_i povie, najviac ako dlho môže počítať program P_{i+1}.)

Ku každému programu Q existuje pekná postupnosť, ktorá ho obsahuje.

Q8 Množina pravdivých tvrdení v našej aritmetike prirodzených čísel so sčítaním a násobením (postavenej na predikátovej logike prvého rádu) je rekurzívne vyčísliteľná.

2 (True or False) and Justify – riešenie [20 bodov]

Q1 TRUE. Ľahko zložíme g napr. kompozíciou z f, *pow* a konštánt.

Q2 TRUE. Keď chceme vypočítať hodnotu $g(n)$, hľadáme i také, že $f(i) = n$. (Takýchto i môže byť viac, nám stačí nájsť jedno ľubovoľné.)

Program pre g vyzerá tak, že postupne počítame hodnoty $f(i)$ pre rastúce i. Keďže f je surjektívna, máme istotu, že časom nájdeme také i, pre ktoré $f(i) = n$. Preto tento program vždy zastane, a teda g je rekurzívna.

Q3 TRUE. Tá istá argumentácia ako v Q2, len hodnoty $f(i)$ musíme rátať paralelne, nie sekvenčne.

Q4 TRUE. Zo samotnej funkcie f ľahko definujeme jeho čiastočnú charakteristickú funkciu $\xi_D(n) = \mathrm{sgn}(1 + f(n))$.

Alebo cez generátory: paralelne počítame f pre všetky možné n. Vždy, keď pre nejaké n výpočet skončí, dáme hodnotu n na výstup.

Q5 TRUE. Tu je výhodnejší pohľad cez generátory: paralelne počítame f pre všetky možné n. Tentokrát ale vždy, keď pre nejaké n výpočet skončí, dáme na výstup hodnotu, ktorú tento výpočet vypočítal.

Q6 TRUE. Je ňou napríklad signum univerzálnej funkcie pre unárne primitívne rekurzívne funkcie.

(Pozor! Napríklad funkcia $f(n) = A(n,n) \bmod 2$ v skutočnosti je primitívne rekurzívna, vieme ju totiž rátať aj bez toho, aby sme spočítali $A(n,n)$. Ale viaceré iné variácie na tému Ackermann fungujú – napr. by som si tipol, že najvýznamnejšia cifra $A(n,n)$ primitívne rekurzívna nebude.)

Q7 FALSE. Pomocou indukcie a vety o primitívne rekurzívnej časovej zložitosti vieme dokázať, že všetky programy v každej peknej postupnosti počítajú len primitívne rekurzívne funkcie. Teda napríklad program Q počítajúci Ackermannovu funkciu nemôžeme v peknej postupnosti stretnúť.

(Pozor! Program počítajúci Busy Beaver funkciu neexistuje, nemôžeme ho teda použiť ako kontrapríklad.)

Q8 FALSE. Na prednáške bol dôkaz, že táto množina je aspoň taká ťažká ako komplement $HALT$-u, lebo vieme v aritmetike pre ľubovoľný program P sformulovať tvrdenie „program P sám na sebe nezastane“.

3 Známa pôda [4+4+7 bodov]

V tejto úlohe sa pýtam na priamo odprednášané veci, ukážte, že im dostatočne rozumiete.

Nech φ_i je efektívne číslovanie unárnych čiastočne rekurzívnych funkcií, pričom φ_0 je nikde nedefinovaná funkcia (teda $\forall x : \varphi_0(x) = \perp$), φ_4 je identita a $\varphi_{4747}(x) =$ (ak x je štvorec tak $\sqrt{x}$ inak $\perp$).
Nech $PRIME$ je množina všetkých prvočísel a nech $HALT = \{n \mid \varphi_n(n) \neq \perp\}$.

a) Dokážte, že $PRIME \leq_m HALT$. V bežnom programovacom jazyku naprogramujte vyhovujúcu redukciu.
b) Rozhodnite a dokážte, či $HALT \leq_m PRIME$.
c) Rozhodnite a dokážte, ktoré z množín $PRIME$ a $HALT$ sú m-úplné.

4 Známa pôda – riešenie [4+4+7 bodov]

4.1 Podúloha a)

Zo zadania vieme, že $4 \in HALT$ a $0 \notin HALT$. Redukcia môže vyzerať nasledovne:

```
def is_prime(n):
    if n<2: return False
    for i in range(2,n):
        if n%i==0: return False
    return True

def redukcia(n):
    if is_prime(n): return 4        # cislo fcie, co sama na sebe zastane
    return 0                        # cislo fcie, co sama na sebe nezastane
```

(Pozor! Redukcia musí byť funkcia, ktorej výpočet vždy skončí! Vstup je číslo, výstup je tiež číslo. Obľúbená chyba bola „redukcia“, ktorá namiesto return 4 zastala a namiesto return 0 sa zacyklila.)

4.2 Podúloha b)

Neplatí to: jednoducho preto, že takáto redukcia by spolu s vyššie uvedenou funkciou `is_prime` tvorila algoritmus rozhodujúci problém zastavenia, a teda $HALT$ by bola rekurzívna.

4.3 Podúloha c)

Kvôli podúlohe b) nemôže $PRIME$ byť m-úplná.
Dôkaz m-úplnosti $HALT$ je v skriptách. (Predstavte si, že sem bol copy&pastenutý.)

5 Formalizmus [5+10 bodov]

V tejto úlohe kladťe hlavný dôraz na presnosť a korektnosť práce s formalizmom z prednášky.

Uvažujme Minského registrový stroj. Program preň je postupnosť očíslovaných inštrukcií `INC x`, `DEC x` a `ZERO x y [z]`.
Uvažujme konštrukciu, ktorá zo stroja so štyrmi počítadlami vyrobí ekvivalentný s dvoma. Formálne definujte, ako je pri tejto konštrukcii reprezentovaný obsah pôvodných počítadiel v nových počítadlách.
Následne napíšte časť programu, ktorá odsimuluje zväčšenie hodnoty v počítadle 2. T.j. napíšte tú časť programu, ktorá sa v novom stroji vykoná vtedy, keď by pôvodný stroj vykonal inštrukciu `INC 2`.

6 Formalizmus – riešenie [5+10 bodov]

Označme hodnoty v pôvodných registroch a, b, c, d. V novom stroji budeme mať vo významných okamihoch v registri 0 hodnotu $2^a 3^b 5^c 7^d$, zatiaľ čo register 1 bude prázdny.
Zväčšenie pôvodného registra 2 (nech je to premenná c) teda zodpovedá vynásobeniu nového registra 0 piatimi. Predpokladáme, že na začiatku obsahuje nový register 1 nulu, na konci ho do takého istého stavu vrátime. Program:

```
1. ZERO 0 5 2
2. DEC 0
3. INC 1
4. ZERO 0 5 2
5. ZERO 1 13 6
6. DEC 1
```

```
 7. INC 0
 8. INC 0
 9. INC 0
10. INC 0
11. INC 0
12. ZERO 1 13 6
```

7 Exkurzia do neznáma [10 bodov]

V tejto úlohe sa pýtam na niečo, čo som priamo neprednášal, a chcem vidieť, ako si s tým poradíte.

Pošlite mi SMS s definíciou čo najrýchlejšie rastúcej funkcie.
(Kto nemá kredit alebo sa mu nechce, môže napísať na papier. Ale samozrejme: limit je 160 znakov!)

8 Exkurzia do neznáma – riešenie [10 bodov]

Tu je môj pokus (spred písomky, má 154 znakov, ukázalo sa, že ostal neprekonaný):

```
Nech B_0 je Busy Beaver pre Minskeho stroje a nech B_{n+1} je to iste pre stroje ktore navyse
maju instrukciu na vypocet B_n. Moja funkcia je f(n)=B_n(n).
```

Teda napríklad keď definujeme B_1, máme meta-Minského stroje s inštrukciami `INC x`, `DEC x`, `ZERO x y [z]` a `BO x`, pričom `BO x` zoberie obsah registra x a nahradí ho hodnotou $B_0(x)$.
Pre takúto množinu strojov môžeme teda opäť $B_1(n)$ definovať ako najväčšie číslo, ktoré vyrobí na prázdnom vstupe stroj, ktorý má najviac n (nových) inštrukcií a v konečnom čase zastane.
Všimnite si, že už B_1 rastie neporovnateľne rýchlejšie, ako čokoľvek, čo viete „naskladať“ z funkcie B_0 pomocou kompozícií, cyklov a rekurzie – teda aj ako napr. $\underbrace{B_0(\ldots(B_0(n))\ldots)}_{B_0(n)-\text{krát}}$.

Vedeli by ste teraz definovať funkciu, ktorá rastie rýchlejšie ako každá z mojich B_i?